Q61372
11/29/00
2 of 2

Patent Office
Canberra

I, LISA TREVERROW, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 4924 for a patent by CSIRO, DIVISION OF MATHEMATICAL AND INFORMATION SCIENCES filed on 31 December 1999.

WITNESS my hand this
Thirteenth day of October 2000

Lisa Treverrow

LISA TREVERROW
TEAM LEADER EXAMINATION
SUPPORT AND SALES

CERTIFIED COPY OF PRIORITY DOCUMENT

P/00/009 28/5/91
Regulation 3.2

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION

Invention title A METHOD AND SYSTEM FOR COMMUNICATION IN THE USENET

The invention is described in the following statement:

A Method and System for Communication in the Usenet

Field of Invention

The present invention relates to Internet information services. In one aspect in particular, the present invention relates to a method and system for communication and / or efficient exchange and storage of binary objects in the Usenet and similar systems. This aspect may be described as "Advanced News Server" (ANS).

Another aspect relates to using metadata to help Usenet users make informed decisions on whether they want or not to download a particular Usenet article.

Background

The Usenet is a worldwide bulletin board system that can be accessed through the Internet or through many online services. The Usenet contains tens of thousands of forums, called newsgroups, that cover many and varied interest groups. The Usenet is used daily by millions of people around the world [1].

Every Usenet message belongs to a newsgroup. Messages are made available to users worldwide by means of the UUCP and NNTP protocols (Unix to Unix Copy Program, and Network News Transport Protocol [2, 3, 4,6], respectively). Individual computing sites appoint somebody to oversee the huge quantity of incoming messages, and to decide how long messages can be kept before they must be removed to make room for new ones. Typically, messages are stored for less than a week. They are made available via a news server.

Users access local newsgroups with a newsreader program. Modern WWW browsers come with a built-in newsreader. A dedicated newsreader program can also be used. The newsreader accesses the local (or remote) News host using the Network News Transfer Protocol (NNTP), enabling one to pull down as many newsgroups and their contents as one desires. If there is no local access to News, there are publicly accessible commercial and free Usenet hosts that can be accessed.

Users sending Usenet messages must address each message to a particular newsgroup [5]. There are newsgroups on subjects ranging from education for the disabled to Star Trek and from environment science to politics in

the former Soviet Union. The quality of the discussion in newsgroups may be excellent, but this is not guaranteed. Some newsgroups have a moderator who scans the messages for the group and decides which ones are appropriate for distribution.

Some of the newsgroups provide a useful source of information and help on technical topics. Users needing to find out about a subject often send questions to the appropriate newsgroup, and an expert somewhere in the world can often supply the answer. Lists of Frequently Asked Questions are compiled and made available periodically in some newsgroups.

The transmission of Usenet news is cooperative [5]. There are places which provide feeds for a fee (e.g. UUNET), but for the large part news transmission is carried out on the basis of peer agreements.

There are two major transport methods, UUCP and NNTP, as previously noted. The first is mainly modem based and involves the normal charges for telephone calls. The second, NNTP, is the primary method for distributing news over the Internet.

With UUCP, news is stored in batches on a site until the neighbor calls to receive the articles, or the feed site happens to call. A list of groups which the neighbor wishes to receive is maintained on the feed site. The Cnews system compresses its batches, which can dramatically reduce the transmission time necessary for a relatively heavy newsfeed.

NNTP, on the other hand, offers a little more latitude with how news is sent. The traditional store-and-forward method (as noted above) is, of course, available. Given the "real-time" nature of the Internet, though, other methods have been devised. Programs now keep constant connections with their news neighbors, sending news nearly instantaneously, and can handle dozens of simultaneous feeds, both incoming and outgoing.

The transmission of a Usenet article is centered around the unique 'Message-ID:' header. When an NNTP site offers an article to a neighbor, it says it has that specific Message ID. If the neighbor finds it hasn't received the article yet, it tells the feed to send it through; this is repeated for each and every article that's waiting for the neighbor. Using unique IDs helps prevent a system from

receiving five copies of an article from each of its five news neighbors, for example.

The Usenet was originally designed for exchange of textual information, but presently the major part of bandwidth and storage resources is consumed by so called "binary" newsgroups that mainly carry binary data. In terms of bytes, the top four newsgroups consume 22% of the entire volume. The top 35 groups consume 50% of the entire volume [7].

In relation to the first aspect, many Internet Service Providers do not service a lot of the binary groups because these binary groups are considered to send the total volume of news soaring. The total news feed is said to be about 25 to 30 Gb a day [8].

If otherwise normal text groups get relatively large volumes of binary objects posted then there is a danger that ISPs will drop them from their news feeds. There are approved cancel 'bots' that remove all messages containing large binary objects from the main news groups. It is the action of those people who cancel and the restraint of the majority of users that helps to keep the newsgroups alive.

The average text message is probably about 2K or less in size (unless it also contains HTML) but a binary object can easily run from 20K to 250K. For many groups a single binary object can equal the entire day's text download.

News articles are stored in news servers to enable users to access them. But this storage brings about another problem, that being the limited availability of storage space. To limit amount of disk space occupied by binary newsgroups, ISPs normally set shorter expiration time limit for binary postings. This helps to save disk space in short term, but users of popular binary news groups compensate for this by re-posting popular binary objects regularly, to ensure their availability. This reduces the effect of the measures taken by ISPs and even makes the situation worse because:

Often a binary object is re-posted by more then one poster and there are several copies of it stored on the server with different messages.

Regular re-posting of large binary objects leads to waste of bandwidth that can be and should be avoided.

Another problem is being caused by violation of the Usenet etiquette by some posters that want as many people as possible to see their messages and to achieve this, they send the messages to many newsgroups. In extreme cases, they send messages to newsgroups that are hardly related to the topic.

A major part of storage and traffic resources is spent because all messages, including binary objects, have to be sent and stored in textual format. There is no compression for textual messages, and binary objects have to be text-encoded. This does not decrease their size. Quite opposite, this increases their size by 33%.

Some attempts have been made in the past to address these problems, but with limited success.

As described above, some ISPs try to reduce expenses caused by handling binary attachments by setting low limit on time that a message with a binary object will spend in the news pool on their server. However, this is not an effective solution because often the same binary object returns re-posted with a new message. This increases news feed traffic and leads to multiple copies of the same object being stored.

News server software that uses UUCP for news feeding (such as the Cnews program) compresses sets of news messages before transferring them. Compression allows to reduce bandwidth requirements, but most of binary data (e.g. images and video) are hard to compress without loss of quality. This means that compression is considered useful when applied to textual data, but not considered useful when applied to most kinds of binary data.

News caching is a popular approach. It has been implemented in Dnews software. This method does not download news messages until a user shows interest in the newsgroup. Once a user has subscribed to a newsgroup, the whole newsgroup is downloaded. This method does not save from expenses caused by duplicated binary objects, however. Also, if the number of users is considerably large, this method is unlikely to provide significant advantage because most of the newsgroups end up downloaded.

There does exist some patent literature related to the problem of storage and exchange of information in an electronic environment, but these disclosures

are also not considered to solve the problem(s) noted above. In particular, there is:

Patent No US 5,771,355 - Title: Transmitting Electronic Mail by Either Reference or Value at File-Replication Points to Minimise Costs. This patent covers technology aimed at improving e-mail delivery in certain conditions. E-mail attachments are delivered by "optimal path". For example, when the path includes intermediary points that make it much longer than the distance from the sender to the receiver, it makes sense to defer sending of attachment until the receiver requests it and, in this case, send attachment directly from the site where it is stored to the receiver.

However, the disclosure does not appear to address the Usenet, nor the duplication problem noted above. Addressing the problem of finding equivalent objects attached to different messages and posted by different users also does not appear to be disclosed.

Patent No US 5,903,723 - Title: Method and Apparatus for Transmitting Electronic Mail Attachments with Attachment References. The disclosure relates to a modified version of the patent discussed above, but it too does not appear to address the issues noted above.

Patent No US 5,813,008 - Title: Single Instance Storage of Information. This patent relates to avoiding storing multiple copies of 'common portions' of information records on a network of storage devices. The disclosure, however, does not relate to the Usenet, but to email. In the email system disclosed, when a user's mailbox is moved to a new server, the single-instance identifiers of the messages in the moved mailbox are compared to a table of single-instance identifiers associated with messages already stored on the new server. Copies are made of only the common portions for which a copy is not already stored on the new server. From this it can be seen that the disclosure relates to avoiding storing multiple copies within a single server, not within the network as a whole. Otherwise they would not have to make copies "of only the common portions for which a copy is not already stored on the new server."

The disclosed method for finding common portions finds only common portions created as a result of modifying the same information item (e.g. e-mail

message). In other words, the common portions are inherited by the items from a common ancestor. However, this does not address problems associated with finding attachments posted by different users independently, and thus, not having any common ancestors that could be traced.

Patent No US 5,815,663 - Title: Distributed Posting System Using an Indirect Reference Protocol. This patent disclosure describes posting marked up messages to news groups. In this system, a message would look like a HTML page with various elements (like images) and links to other pages or messages. The patent describes two ways to give access to the page elements. The first one is to send them with the message as attachments. The second one is to provide a URL-like references to the elements.

Again, this patent disclosure is not considered to address the problem with attachments posted by different users independently, or even avoiding storing same objects posted as attachments by the same user.

Patent No US 5,815,663 - Title: Method and Apparatus for Identifying Duplicate Data Messages in a Communication System. This patent disclosure is considered directed at how to determine whether one message is a copy of another message in an environment where errors are very frequent. In the Usenet, however, the environment is relatively error free, and thus the problems addressed in this disclosure are not considered relevant to the problems of the present invention.

Publication No 05316143 (Japanese) - Title: Electronic Mail Processor and Method Therefor. In this disclosure, instead of sending e-mail message to all destination mailboxes, it is suggested to send only its id and to keep the message in a central repository until requested. Again, it appears unrelated to the Usenet.

In relation to the second aspect, as noted above, given large average size of binary objects, pollution of binary newsgroups by spam and slow speed of downloading via modem lines, it is very important to help users to make better decisions on whether to download a particular binary object. Because, if this decision is wrong, they spend resources (their own time, on-line time, traffic) on downloading an object that they will discard right after downloading.

In decentralised, anarchic systems like Usenet it is important to provide people with better means of orientation, filtering spam and selection of quality items. Better descriptions of resources, based on use of metadata can help to solve this problem.

Some attempts have been made in the past to address the problem, but with limited success.

Currently, almost the only description of an article is its subject. This way of describing information items is more or less adequate for textual messages that contain text discussing the subject. For multimedia items, one-line description can hardly be adequate. Normally, subject contains name of the collection or short description of the multimedia item, name of file, number of the part and total number of parts (such as "Persian kitten cats123.jpg (1/1) 35567 bytes"). This format is often used, but many multimedia postings do not have even that. Often subject lines are quite meaningless, eg. "My loved kittens".

Images represent a significant part of multimedia objects posted on the Usenet. It can be noticed that users posting large collections of images (tens or hundreds of them) often post so called "indices" - images that contain thumbnails (small copies) of images posted in the collection. This gives to the downloaders the opportunity to download an "index" image and get a better idea about the images posted in the collection, make better informed decisions whether to download a particular image and thus save downloading time and money spent on the Internet session.

This illustrates an approach that uses a different kind of description of an information item. Naturally, a little copy of image is a better description of it than a subject line.

This method allows users to save downloading time, but it is quite labor intensive. To download a set of selected images from a posted collection, the user has to perform the following steps:

- Locate collection articles;
- Locate collection indices;
- Download collection indices;
- View collection indices and memorise (write down) names of wanted files;

- Locate articles carrying the wanted files and either mark them for background downloading or download them instantly;

As it can be seen, this may require considerable time and effort.

MIME standard [9, 10] allows to incorporate references in bodies of the messages and refer to other objects accessible using some protocol specified by the reference. This feature can be used to refer to binary objects from their descriptions. However, there are two problems here:

First, message containing metadata information (descriptions) must be recognisable by its header. Otherwise, the client will have to download all message bodies to find descriptions. This defies the purpose of the effort and is clearly unpractical.

Second, at the time of message construction, the client does not know what message ID will be assigned to the target message of the reference. So, using MIME references mechanism is not straightforward.

It is an object of the present invention to alleviate at least one problem associated with the prior art. In particular, one aspect of the present invention seeks to address problems associated with the Usenet and problem of finding the same object attached to different messages and posted by different users that also does not appear to be disclosed. Another aspect of the present invention seeks to provide a better way of describing multimedia items.

Summary of Invention

First Aspect

A first aspect of the present invention provides a method of alleviating storage of duplicate binary objects, in a Usenet system, the method including:

1. allocating an identifier, such as UBOI or RUBOI to a first binary object,
2. determining whether the system has already stored a second binary objectequivalent to the first binary object, and
3. storing the first binary object if the result of step 2 is negative.

Preferably, the method further includes

4. substituting in the message the first binary object by a reference to it and storing the message.

Preferably, if the result of step 2 is positive, the message is stored together with a reference to the second binary object.

The present invention provides also a method of identifying, in a Usenet system, duplicated binary objects, the method including:

1. making available information identifying a first binary object,
2. determining whether the system has already stored a second binary objectequivalent to the first binary object, and
3. determining that there is a duplication of binary objects if the second object is equivalent to the first object.

Preferably, the system transfers messages only with binary objects that are not equivalent.

Other features of this aspect of the present invention are outlined in the claims.

The present aspect is considered to address the problem of reducing the cost of transferring and storing Usenet messages that include large binary objects, such as images, sound, video, executable code, etc. The system we are inventing is based on the existing Usenet standards and architecture, in particular, the NNTP protocol although other functionally similar protocols can be used in a similar way.

The present aspect is based on the recognition that there are a significant number of duplicates among the posted binary objects:

1) That have been posted to the same group simultaneously by different posters;
2) That have been posted to different groups simultaneously by the same or different posters;
3) That have been posted recently and then re-posted.

The present aspect helps to identify the duplicates and to avoid storing and transferring multiple copies of the same binary object.

The above methods are equally applicable to RUBOI \as herein disclosed.

In general, a Universal Binary Object Identifier can be considered a sequence of bytes, or information, that is assigned to binary object in order to identify it, and that has the following properties:

1. It is smaller than the object it is identifying;
2. The probability of two different objects having the same identifier is insignificantly low for practical purposes.

It is a function of the object's content and properties. This means, that, having an object, it is possible to construct UBOI for it using a particular algorithm. For example, we describe building UBOIs by calculating CRC32 code of the object and its size.

In general, a Reliable Universal Binary Object Identifier can be considered a sequence of bytes, or information, that is assigned to binary object in order to identify it, and that has the following properties:

1. It is smaller than the object it is identifying;
2. Two different objects always have different identifiers.

In general, a "binary object" is a form of data or information communicable in electronic format. In one form, unlike that of textual objects, their natural format of presentation and/or processing is not textual. Examples of binary objects: images, executable code, video files, sound files, even compressed text.

In general, by the term 'Usenet', we mean the Usenet or any information system based on the following principles:

1. There are a number of interacting servers that store information items,
2. The information items are exchanged (preferably automatically) between the servers and replicated on them.
3. Users (or client programs) typically access the system via a small number of servers of their choice.
4. Users (or client programs) can post (contribute) information items to the system and/or retrieve items, including ones contributed by other people.

If a news server already has the object, the invention considers that there is no need to transfer and store a new copy of it. A single copy can be shared among all messages that have this object included. Only a reference to the shared object has to be stored with each message.

The present invention will identify binary objects by their unique parameters, such as, but not limited to, CRC32 code plus file size. Thus, if two messages have attached binary objects that have identical CRC32 codes and

size, the present invention assumes that the binary objects are really the same and stores only one copy of them. It is considered that the probability of two different objects having these two parameters identical is very small, practically zero. In case if this level of reliability is insufficient, one of reliable methods of assigning binary objects identifiers described below can be used.

For the transfer of messages between two ANS-complaint (i.e. supporting this invention) news servers, the present invention will offer attachment identification information along with the message identification information (Message ID) to the receiving server to make the decision whether the attached binary object has to be transferred. If the receiving server has a copy of the binary object already, it may decide that no transfer of the binary attachment is necessary and accept only the textual part of the message.

When the sending server is not ANS-complaint (i.e. a server that does not support this invention) and does not offer object identification information, the receiving server will accept the beginning of the binary object (that typically includes file name and a part of the body) and make decision based on this incomplete information. For example, if it has already a binary object that has the same file name and starts with the same sequence of bytes, it is very probable that it is the same object as the one being received. At this stage the receiving server may decide to interrupt receiving the object.

The same technique will be applied to downloading of binary objects by ANS-complaint news reader programs (news clients). Binary object identification information will be included in message headers that clients will receive before downloading the message body. A client can maintain a database of descriptions of binary objects that it has downloaded before. Based on the information in this database and the attachment identification information in the message header, the client can advice the user whether this binary object has been downloaded before, and thus help to avoid downloading duplicates.

The advantages of the present invention include:

1. Relatively economic use of bandwidth and hard disk space because duplicated binary objects are shared between messages and only one copy is transferred and stored.

2. Increased performance of software due to dealing with smaller amount of data (transferring, saving, reading it etc.)

3. Flexibility to configure a system to show the optimal performance in a wide range of circumstances (see below). The present method allows configuration of software to save bandwidth at expense of disk space or vice verse, save disk space at expense of bandwidth, or adapt to any predetermined/selected requirement in the range between these two extreme cases.

4. Decreased traffic expenses because the invention does not use textual encoding of binary objects when transferring them.

Second Aspect

A second aspect of the present invention provides a method of coordinating the identification of objects with their associated descriptions (metadata) in a newsgroup of the Usenet,

The method including the steps of:

Generating a first tag, the first tag being readable in a manner for the purposes of identifying a description,

Attaching the first tag to a message containing the description,

Determine from the first tag, a second tag, the second tag being adapted to identify an object,

Attaching the second tag to the message containing the object,

Posting the messages.

There is also provided a method of downloading messages from the Usenet, the method including the steps of:

Receiving headers of messages available for downloading,

Scanning these headers to identify which messages contain descriptions,

Downloading the messages containing descriptions,

Representing the descriptions to the user to make a decision regarding the downloading of associated objects,

if the user wants to download an associated object,

reading a first tag associated with the description, generating a second tag adapted to identify an object,

Scanning the headers in order to locate a tag equivalent to the second tag, and

downloading the message having the located tag.

Preferably the second tag is the same as the first tag.

This aspect of invention is based on an automatic way of providing a metadata description for every multimedia item and associating metadata descriptions with the information items when the information is being presented to the user during selection process.

It allows to establish connection between messages containing information items and messages containing descriptions (metadata) of the information items. It does this by inserting special fields (tags) in message headers at posting stage. So, at downloading stage, client program, having downloaded message headers, can recognise metadata messages by these special tags in their headers, download the metadata messages, and thus obtain information describing other messages and use this information to better represent these messages to the user.

The method of the second aspect preferably includes two stages.

Stage 1

A collection of multimedia items and its description is posted by poster's client. Description of the collection is an article or a set of articles containing a metadata item for every item of the collection. There are different ways to indicate association between the multimedia items and corresponding metadata items. To our point, the most practical of them is to provide certain tags in headers of the item message and MIME headers of the attachment containing the metadata item.

For example, message carrying file cats123.jpg could contain a header like this:

X-meta-tag: <unique-object-id-1-of-cats123.jpg>

In a message carrying multiple attachments, there would be several such headers, one for each object attached. The correspondent attachment (metadata item) in the collection description article would contain the following string in one of its MIME headers:

X-meta-tag: <unique-object-id-1-of-cats123.jpg>

Thus, provided that we have access to headers of collection articles and to the body of collection description article, we can match descriptions to the articles based on identity of the string in correspondent X-meta-tag fields.

To identify collection description articles, we would add a special header to them, such as

X-metadata: yes

Stage 2

The downloader's client downloads headers of all new articles in the newsgroup. It identifies collection description articles, automatically downloads them (if this is allowed by the user) and uses the found metadata objects (such as thumbnails) to represent the available articles to the user for selection.

The user considers presented information and either marks some of the articles to download in batch mode or double clicks on them to download them immediately.

This approach significantly simplifies for users the process of selecting and downloading of multimedia items. For example, in case of images, user sees a set of small images (thumbnails) on the screen. Each of these images represents a "real" large image. To download real images, the user just has to double click on a thumbnail or select a few of them and then start batch downloading.

This kind of interaction is widely used on the Web, but the underlying protocol (HTTP) is different. Our invention makes it possible to achieve the same level of convenience when working with the Usenet-like systems.

The advantages of the present invention include:

A better representation of available articles during selection stage. This allows to avoid downloading multimedia objects that are unwanted and will be discarded later anyway.

This invention provides a general, flexible and easily extensible way of associating of additional information with articles and using this information when required.

Embodiments of the present invention will now be described with reference to the accompanying drawings, in which:

Figure 1 illustrates schematically differences between the first inventive aspect and the prior art.

Figure 2 illustrates schematically a 1st method applicable to the first aspect that can be used to reliably identify binary attachments.

Figure 3 illustrates schematically a 2nd method applicable to the first aspect that can be used to reliably identify binary attachments.

Figure 4 illustrates schematically a 3rd method applicable to the first aspect that can be used to reliably identify binary attachments.

1.1 First Aspect: First Embodiment

In this embodiment, this invention can be implemented by changing the way news server stores messages in the database and introducing extended analogues of ARTICLE, BODY, IHAVE, NEWNEWS, and POST commands of the NNTP protocol. We will call them XARTICLE, XBODY, XIHAVE, XNEWNEWS and XPOST respectively.

This embodiment is not the only form in which the invention can be performed, and thus the invention should not be limited to the embodiment disclosed.

In terms of this invention, the server will store message bodies and binary attachments separately. Only a reference to the binary attachment will be stored with the message. On the other side, with each binary object an integer number will be stored with the value equal to the number of messages referring to this binary object. If this number is zero, no messages in the server's database have this object as a binary attachment and the object can be safely removed. However, it can be considered keeping "unattached" objects in the database for a while, just in case that they will be re-posted with a new message soon.

Fig. 1 illustrates transition from storing binary attachments 1 in messages 2 to storing binary attachments 1A, 1B, etc separately and providing references 3 from the corresponding messages 2A, 2B, etc to their corresponding binary attachments. There are two different binary attachments in the picture, each is shared among 3-4 messages. We need to store only one copy of each attachment in the case of the present invention. The messages 4 do not have corresponding or attached binary objects.

1.1.1 Extended Commands

The present invention introduces Universal Binary Object Identifier – a code that describes and uniquely identifies a binary object. This code is constructed with the purpose of reliably identifying binary objects. As mentioned above, a pair consisting of a CRC32 checksum and byte size of the object is considered to be reliable enough identifier for the purpose of this invention. If the probability of two objects having same size and CRC32 code is not low enough, other way of constructing UBOI can be chosen to make this probability as low as desired. For example, we can base UBOI on two CRC32 codes, where the first one is for the first half of the object, and the second one is for the second half of the object.

A full description of NNTP protocol is available in [2]. In the text below we will only define extended versions of a few commands that we need for the purpose of our invention.

1.1.2 XARTICLE Command

XARTICLE *<message-id> ["*"| <UBOI$_{k1}$>, <UBOI$_{k2}$>,...]*

Send the header, a blank line, then the body (text) of the specified article with binary attachments replaced by their UBOIs. Then send all binary attachments if symbol "*" follows the message-id or only those binary attachments that correspond to UBOIs listed in the XARTICLE command.

Each binary attachment is sent as a sequence <headers \n\n length \n\n bytes \n\n> where headers is a set of ASCII text lines separated by new line (\n) characters. Length is a numeric value of the length of the binary object. Bytes are bytes of the binary object.

Message-id is the message id of an article as shown in that article's header. It is anticipated that the client will obtain the message-id and UBOIs from a list provided by the NEWNEWS command, from references contained within another article, or from the message-id provided in the response to some other commands.

1.1.3 XBODY Command

XBODY command is identical to the XARTICLE command except that it does not send the header lines of the message.

1.1.4 XIHAVE Command

XIHAVE <message-id> *[<UBOI$_1$>, <UBOI$_2$>,...]*

The XIHAVE command informs the server that the client has an article whose id is <message-id> and that includes the listed binary objects. If the server desires a copy of that article, it will return a response instructing the client to send the entire article. If the server does not want the article (if, for example, the server already has a copy of it), a response indicating that the article is not wanted will be returned.

Responses

235 article transferred ok

335 *["*"| <UBOI$_{k1}$>, <UBOI$_{k2}$>,...]* send the article with the listed binary attachments

435 article not wanted - do not send it

436 transfer failed - try again later

437 article rejected - do not try again

If transmission of the article is requested, the client should send the article, including header, body, and requested binary objects in the manner specified for text transmission from the server (see XARTICLE command above). A response code indicating success or failure of the transferral of the article will be returned.

1.1.5 XNEWNEWS Command

XNEWNEWS newsgroups date time [GMT] [<distribution>]

For a full description of parameters of this command see description of the NEWNEWS command in [2]. XNEWNEWS sends a list of message-ids and UBOIs of articles and their attachments posted or received to the specified newsgroups since "date". It differs from the NEWNEWS command only by including UBOIs after message-ids. The format of the listing will be one message-id per line, as though text were being sent, followed by UBOIs of its binary attachments. A single line consisting solely of one period followed by CR-LF will terminate the list.

1.1.6 XPOST Command

XPOST command is similar to XIHAVE command, but it does not include message-id. It does include UBOIs, however, and the server may decide that binary attachments do not have to be transmitted.

Example of a news transfer session using NNTP protocol and our extensions

Using the news server to distribute news between systems.

Server: (listens at TCP port 119)

Client: (requests connection on TCP port 119)

Server: 201 Foobar NNTP server ready (no posting)

(client asks for new newsgroups since 2 am, May 15, 1985)

Client: EWGROUPS 850515 020000

Server: 235 New newsgroups since 850515 follow

Server: net.fluff

Server: net.lint

Server: .

(client asks for new news articles since 2 am, May 15, 1985)

Client: XNEWNEWS * 850515 020000

Server: 230 New news since 850515 020000 follows

(following article does not have a binary attachment)

Server: <1772@foo.UUCP>

(following article does not has a binary attachment with length 230543 bytes and CRC32 code 2938464828)

Server: <87623@baz.UUCP> <230543 2938464828>

(following article has two binary attachments, the first of them the same as in the previous message)

Server: <17872@GOLD.CSNET> <230543 2938464828> <298799 6534821>

...

Server: .

(client asks for article <1772@foo.UUCP>)

Client: XARTICLE <1772@foo.UUCP>

Server: 220 <1772@foo.UUCP> All of article follows

Server: (sends entire message)

Server: .

(client asks for article <87623@baz.UUCP> and its binary attachment)

Client: XARTICLE <87623@baz.UUCP> <230543 2938464828>

Server: 220 <87623@baz.UUCP> The article and its attachment follow

Server: (sends message body)

Server: .

Server: (sends binary attachment)

(client asks for article <17872@GOLD.CSNET> and only the second of its attachments because it already has the first one)

Client: XARTICLE <17872@GOLD.CSNET> <298799 6534821>

Server: 220 <17872@GOLD.CSNET> The article and its attachment follow

Server: (sends message body)

Server: .

Server: (sends requested binary attachment)

(client offers an article it has received recently)

Client: XIHAVE <4105@ucbvax.ARPA>

Server: 435 Already seen that one, where you been?

(client offers another article)

Client: XIHAVE <4106@ucbvax.ARPA> <378699 666237> <126789 76367>

Server: 335 * Send the article and all its attachments

Client: (sends textual body of the article)

Client: .

Client: (sends first binary attachment)

Client: (sends second binary attachment)

Server: 235 Article transferred successfully. Thanks.

Client: QUIT

Server: 205 Foobar NNTP server bids you farewell.

1.2 First Aspect: Second Embodiment

1.2.1 Global References and Binary Servers

As described above, the present invention stores binary attachments separately and stores only a reference to the binary attachment with the message. If we make this reference global, i.e. it can point to a binary object on another server, it makes it unnecessary to download the attachment until a user had requested it. More than this, user's client program can be referred to the actual server that has this binary object stored, so that it can download the binary object from that server. Thus, there is no need for the local news server to keep the attachment at all. This role can be appointed to a dedicated server that stores and serves binary objects to a sharing community of news servers.

This architecture of the system does make it relatively more complicated to determine that there are no references to a particular binary object in order to delete it, as references now can be global. However a heuristic criterion based on use pattern is available. If there are no requests for the object for a considerable time interval, it means that it can be safely deleted because, even if the referring messages have not been removed, users are not interested in this object.

Using global references, we can save local hard drive space at expense of global traffic. Storing all binary attachments locally, we can save global traffic at expense of the hard drive space. These are two extreme strategies. The optimal strategy is somewhere between them. It makes sense to store popular binary objects locally (cache them) to minimise global traffic, and the rest of binary objects may be stored on binary servers and referred to by global references.

A 'global' system can be implemented in accordance with the way as it has been described in the first embodiment, with minor changes:

1. store and transmit with each message global references to its binary attachments,

2. introduce a special command that lets to retrieve binary attachment only, without any regard to a particular message. We will call this command XBINARY. Its syntax is XBINARY *<UBOI>*. When a server receives this command, it will return success code followed by the binary object identified by the UBOI or error code if can not send the object.

1.3 First Aspect: Reliable Methods of Identification of Binary Objects – Third Embodiment

No matter how small, there is a probability that two different binary objects will have identical UBOIs. In case it proves to be important to avoid this occurrence, the present invention offers a number of reliable methods of attachment identification. These methods offer reliability at a cost of a small resource overhead. Please note that these methods are only concerned with assignment of reliable identifiers (that can be used instead/together with UBOIs) to binary objects. Storage and exchange of binary objects are implemented in a way described above in first or second embodiments. The syntax and semantics of the introduced protocol commands must be adjusted correspondingly.

The present invention introduces RUBOI - Reliable Unique Binary Object Identifier. The difference between RUBOI and UBOI is that it is more likely that different binary objects have different RUBOIs.

1.3.1 Method A. Identification Request Broadcast

The suggested method is based on requesting of attachment identification information from other Usenet servers. We describe this method as a sequence of numbered steps below.

1. Server 1 receives a message containing a binary attachment that does not have a RUBOI assigned.
2. Server 1 builds UBOI for this attachment and checks if it has other attachments with this UBOI in its storage.
3. If there are such objects, Server 1 compares them to the new one byte-to-byte. If any of the old objects is identical to the new one the server uses its RUBOI. Thus, the attachment has been identified. Go to step 11.
4. If no identical objects found, Server 1 issues a request (system message) containing the UBOI of the new object and RUBOIs of the objects that have been compared to the new object, and posts this request in the Usenet.
5. Upon receiving this request, other servers check their sets of stored binary attachments.

6. If any server finds a binary object that has identical UBOI, and not listed in the request message, it responds with RUBOIs that have not been listed in the request message.

7. If after a pre-set waiting time Server 1 does not receive any messages, it assumes that no other objects with identical UBOI exist, and generates or obtains from a third party a new RUBOI for the new object. Go to Step 10.

8. If Server 1 receives any response messages, it chooses a set of servers that covers all RUBOIs that the new object has not been compared to, and sends the new object to these servers (preferably) or requests binary objects from them for comparison.

9. They compare the new object to their objects with the same UBOI and respond with RUBOI of the identical object, if found. In this case Server 1 uses the found RUBOI. Go to Step 11.

10. A simple method can be used to generate a new RUBOI. For example, RUBOI may be a string containing host and domain names of the Server 1, day and time stamp, and sequential number of the binary object from the start of the day. Alternatively, a new RUBOI can be obtained from a special server (a third party server that is authorised to generate and issue new RUBOIs).

11. End of work.

1.3.2 Method B. Recognition Event Broadcast

This method is based on broadcasting object equivalence information in the Usenet. Initially, every binary object that does not have a RUBOI is assigned a new RUBOI, unless the server that receives it, has this object already and recognises it. Then the server feeds this object to other servers. When any server establishes a fact (e.g. by comparison) that two identical objects have different RUBOIs RUBOI1 and RUBOI2, it posts a system message that notifies other servers that RUBOI1 is equivalent to RUBOI2. We describe this method as a sequence of numbered steps below.

1. Server 1 receives a message containing a binary attachment that does not have a RUBOI assigned, or has a new RUBOI suggested by the client.

2. Server 1 looks for an identical object in its storage. If any of the old objects is identical to the new one, the server uses its RUBOI. Go to Step 8.
3. If no identical objects found, Server 1 generates a new RUBOI for the object (or uses the one suggested by the client that posted the message). A simple method can be used to generate a new RUBOI. For example, RUBOI may be a string containing host and domain names of the Server 1, day and time stamp, and sequential number of the binary object from the start of the day. Alternatively, a new RUBOI can be obtained from a special server (a third party server that is authorised to generate and issue new RUBOIs).
4. Server 1 feeds the object with new RUBOI1 to the servers it is feeding.
5. After receiving the object, every Server 2 looks in its storage for an identical object.
6. If an object found that is identical, but has a different RUBOI2, Server 2 posts a system message that says that RUBOI1 is equivalent to RUBOI2. All servers that receive this message, can use this information later when handling new objects.
7. Steps 5 and 6 are repeated by every server when receiving the new binary object.
8. End of work.

1.3.3 Method C. Centralised Identification

This method is based on use of a central server that has the largest collection of binary objects in the Usenet. It is important (but not critical) that this server has binary object if any other news server has it. This rule is important to provide effective identification of binary objects. (If it is not 100% true, the system will still work, but different RUBOIs will be assigned to some identical binary objects. This will result in decreased efficiency.) We will call this "central identification authority" server Server 0. We describe this method as a sequence of numbered steps below.

1. Server 1 receives a message containing a binary object that does not have a RUBOI assigned or has one suggested by the client that has posted the message.

2. Server 1 checks if it has an identical binary object in its storage.

3. If any of the old objects is identical to the new one, the server uses its RUBOI1. Go to Step 6.

4. If no identical objects found, Server 1 sends the new object to Server 0 for identification. Server 0 looks in its collection for identical objects. If any found, Server 0 sends its RUBOI1 to Server 1 to use for the new object. Go to Step 6.

5. If no identical objects found, Server 1 generates a new RUBOI1 for the object or uses the one suggested by the client. A simple method can be used to generate a new RUBOI. For example, RUBOI1 may be a string containing host and domain names of the Server 1, day and time stamp, and sequential number of the binary object from the start of the day. Alternatively, a new RUBOI can be obtained from a special server (a third party server that is authorised to generate and issue new RUBOIs).

6. Server 1 feeds the object with RUBOI1 to the servers it is feeding.

7. End of work.

1.3.4 Method D. Using Multiple Reliable Identifiers

This method is relatively simple. Each server in the path of the message containing a binary object adds to the header the RUBOI of this object if an identical object already exists in the collection of the server and its RUBOI is different from those that are already in the message header. Thus, the message will have in its header multiple identifiers for the carried binary object.

When this message is being offered to any server, it rejects the binary object if it has a binary object known by any one of the RUBOIs in the message header.

2.0 Second Aspect

Practical implementation of this invention does not require changing of involved standards, such as NNTP, MIME etc. It only requires modification of posting and downloading news clients so that they would add some extra information to messages' and MIME encoded objects' headers during the posting stage and could interpret this information during the downloading stage.

To describe how the system works, we will take as a base work of a standard newsreader e.g. Netscape newsreader that is a part of Netscape Communicator package, Version 4.06. Those skilled in the art are familiar with use of a typical news client. We will describe how the client works in our embodiment. To do this, we will describe what it does differently or additionally to the Netscape news client.

There are two tasks that are performed differently: posting and representing. We will describe each of them.

2.0.1 Posting

The task is to post a collection of one or more multimedia objects. The client does it as normally, with only one difference: if it detects that a message to be posted contains a multimedia object(s), it generates one header for each object and inserts it in the head of the message. In this embodiment, the format of this header is as follows:

X-meta-tag: '<'<CRC32 of the object>-<size of the object>-<time stamp>'>'

Where

CRC32 of the object is a numeric CRC32 code of the object;

Size of the object is number of bytes in the object;

Time stamp is time when the header was generated, with milliseconds.

After this the client creates a metadata description item for each multimedia object in the message and temporarily stores it locally with a tag corresponding to the string in the X-meta-tag header.

The client automatically creates and posts metadata description messages in one or more (this may be controlled by configuration parameters of the client) of the following events:

1. At the end of the session;
2. Every time when the volume of stored metadata items exceeds some threshold;
3. At regular time intervals;
4. By explicit user request.

The temporarily stored metadata description items that have not been posted before are posted in such messages and then deleted. Each metadata

description message is a normal news message containing a set of multimedia objects that are metadata description items of the multimedia objects posted before.

Each metadata description message contains a header in format:

X-metadata: yes

This header allows clients to recognise such messages and download them to present metadata to users for selection.

Each metadata object is MIME encoded and its encoding contains a Content-Description header in format:

Content-Description: "X-meta-tag: '<'<CRC32>-<size>-<time stamp>'>'"

Where the CRC32, size and time stamp values are the same as in the X-meta-tag header of the message that includes the object described by this metadata object.

Example.

First message:

From: catlover@cats.society.org

Newsgroups: alt.binaries.nospam.cats.sleeping

Subject: Pajama Party! Day 2 by popular demand! - 090pjp.jpg (1/1)

Date: 14 Jul 1999 02:42:34 GMT

X-meta-tag: <098283278219-29875-19990714024234123>

Organization: Cats Society Inc.

Lines: 424

<message body including the first binary object>

Second message:

From: catlover@cats.society.org

Newsgroups: alt.binaries.nospam.cats.sleeping

Subject: Pajama Party! Day 2 by popular demand! - 091pjp.jpg (1/1)

Date: 14 Jul 1999 02:45:28 GMT

X-meta-tag: <98273028763-32954-19990714024528265>

Organization: Cats Society Inc.

Lines: 487

<message body including the second binary object>

Metadata description message:

From: catlover@cats.society.org

Newsgroups: alt.binaries.nospam.cats.sleeping

Subject: Collection description message

Date: 14 Jul 1999 03:15:20 GMT

X-metadata: yes

Organization: Cats Society Inc.

Lines: 96

MIME-Version: 1.0

Content-Type: multipart/mixed;

boundary="----------5C18B558FFD309376B5A78B9"

This is a multi-part message in MIME format.

----------5C18B558FFD309376B5A78B9

Content-Type: image/jpeg; name="thumbnail-090pjp.jpg"

Content-Transfer-Encoding: base64

Content-Disposition: inline; filename="thumbnail-090pjp.jpg"

Content-Description: "X-meta-info: <098283278219-29875-19990714024234123>"

<thumbnail of the image 090pjp.jpg>

----------5C18B558FFD309376B5A78B9

Content-Type: image/jpeg; name="thumbnail-091pjp.jpg"

Content-Transfer-Encoding: base64

Content-Disposition: inline; filename="thumbnail-091pjp.jpg"

Content-Description: "X-meta-info: <98273028763-32954-19990714024528265>"

<thumbnail of the image 091pjp.jpg>

----------5C18B558FFD309376B5A78B9--

2.0.2 Representing

The task is to represent available news articles to the end user using available metadata to make a better representation. E.g., normally, only such information as subject, size, poster, date and time of posting is represented about

each article, but for multimedia objects this is clearly not enough. If an image thumbnail is available for image that is contained in article, this thumbnail should be found and used for article representation because in most cases it describes the image better than words of the subject line.

The client accomplishes this task in the following way. The client downloads heads of available news articles as normally. It searches the heads to find ones that contain header "X-metadata: yes". When such header is found, the client automatically downloads the message, parses it (as normally for MIME formatted messages), extracts metadata description items and temporarily stores them with the tags that are found in their "Content-Description" headers.

When building a list of available articles for the user to select from, the client checks each article head whether it contains an "X-meta-tag" header. If yes, the client searches for a stored metadata item that has a correspondent "X-meta-tag" stored with it.

If a correspondent metadata item found, the client uses it to represent the article it relates to. For example, an image thumbnail is used to represent an article that contains the image, a movie clip can be used in representation of an article that contains a movie attached etc.

The user than can make a better informed downloading decision if they have better described articles to select from.

Once selected, the articles are downloaded and presented in a normal way.

REFERENCES

[1] Usernet news. http://sunsite.mus.edu.sg/pub/zen/zen-1.0 6.html.

[2] Brian Kantor, Phil Lapsley, Network News Transfer Protocol. RFC 977, February 1986. http://www.freesoft.org/CIE/RFC/Orig/rfc977.txt

[3] Mark R. Horton. Standard for Interchange of USENET Messages. RFC 850, February 1983. http://www.freesoft.org/CIE/RFC/Orig/rfc850.txt

[4] Standard for the Format of ARPA Internet Text Messages. Revision by David H. Crocker, RFC 822, August 1982. http://www. freesoft.org/CIE/RFC/Orig/rfc822.txt

[5] Usenet news. http://www.terena.nl/libr/gnrt/group/usenet.html.

[6] M. Horton, R. Adams. Standard for Interchange of Internet Messages. RFC 1036, December 1987. http://www.freesoft.org/CIE/RFC/Orig/rfc1036.txt.

[7] Feeder Trace Analysis. http://www.research.digital.com/wrl/projects/newsbench/usenet-html/node14.html.

[8] Article found in DejaNews archive. Posted on 1999//04/19 by David CL Francis flight@nospam.demon.co.uk

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A method of alleviating storage of duplicate binary objects, in a Usenet system, the method including:

a. allocating an identifier, such as UBOI or RUBOI to a first binary object,

b. determining whether the system has already stored a second binary objectequivalent to the first binary object, and

c. storing the first binary object if the result of step 2 is negative.

2. A method as claimed in claim 1, further including the step of

d. substituting in the message the first binary object by a reference to it and storing the message.

3. A method as claimed in claim 1 or 2, wherein the binary object is text or encoded in text, compressed or uncompressed.

4. A method as claimed in claim 1, 2 or 3, wherein, if the result of step b is positive, the message is stored together with a reference to the second binary object.

5. A method as claimed in claim 1, 2, 3 or 4, in which the determining step b is executed via NNTP protocol.

6. In the Usenet system, an identifier, such as a UBOI or RUBOI.

7. An identifier as claimed in claim 6, wherein the identifier includes a checksum and byte size identifier, an extended analogue.

8. An identifier as claimed in claim 7, wherein the identifier includes CRC32 checksum and an indicator of size of the object.

9. An identifier as claimed in claim 7, wherein the identifier includes a combination of a number of CRC32 codes or checksums.

10. A Usenet adapted to operate in accordance with any one of claims 1 to 5.

11. A Usenet including an identifier as claimed in any one of claims 6 to 9.

12. A Usenet as claimed in claim 10 or 11, adapted to operatively respond to any one or a combination of commands described above as XARTICLE, XBODY, XHAVE, XNEWSNEWS, XBINARY and / or XPOST.

13. A method of operating a Usenet in accordance with any one or a combination of commands XARTICLE, XBODY, XHAVE, XNEWSNEWS, XBINARY and / or XPOST as herein disclosed.

14. A method of transferring messages between at least two ANS-complaint news servers, being a receiving server and a sending server, the method including the steps of:

forwarding attachment identification information along with the message identification information (Message ID) to a receiving server,

the receiving server determining whether the attached binary object is to be transferred in accordance with establishing whether the receiving server has a copy of the binary object already, and

if the receiving server does have already a copy of the binary object, indicating to the sender server that no transfer of the binary attachment is necessary but that transferring only the textual part of the message is required.

15. A method of transferring messages between a first ANS-compliant server and a second non-ANS-complaint server which does not offer object identification information, the method including the steps of:

accepting a beginning portion of the binary object by the first server,

determining, based on the beginning portion, whether the first server already has stored a copy of the binary object by comparing the beginning portion with other binary objects already stored, and

if the determination is that a binary object is already stored, requesting the second server to send the textual part of the message.

16. A method as claimed in claim 15, in which the beginning portion includes file name and a portion of the body.

17. A method as claimed in claim 15, in which the beginning portion includes the whole body of the binary object.

18. A method as claimed in claim 15, in which the first server is a receiving server, and the second server is a sending server.

19. A method as claimed in any one of claims 14 to 17, in which when only the textual part of the message is sent, the textual part is stored together with a reference to the already stored binary object.

20. A Usenet adapted to operate in accordance with any one of claims 13 to 19.

21. In a Usenet system, a Reliable Universal Binary Object Identifier (RUBOI).

22. A method of assigning RUBOIs to binary objects, including the steps of:

server that received a binary object that does not have a RUBOI assigned broadcasting in an identification request UBOI of the object and RUBOIs of objects that are known to have identical UBOIs, but are different,

servers having objects with this UBOI and different RUBOIs responding by sending the RUBOIs to the first server,

first server submitting the object to the servers for comparison and identification,

in case of failed identification, generating and assigning of a new RUBOI to the object,

in case of successful identification, assigning existing RUBOI to the object.

23. A method of assigning RUBOIs to binary objects, including the steps of:

server that received a binary object that does not have a RUBOI assigned generating a new RUBOI and assigning it to the object,

server that receives an object that is, in fact, identical to an object with a different RUBOI, broadcasting the fact of equivalence of their RUBOIs,

all other servers remembering the fact of equivalence and using it when making decisions based on comparing RUBOIs.

24. A method of assigning RUBOIs to binary objects, including the steps of:

server that received a binary object that does not have a RUBOI assigned submitting the object to a central "authority" server for identification,

if the authority server can not identify the object, it generates a new RUBOI, assigns it to the object and sends back to the first server,

if the authority server can identify the object, it sends its RUBOI to the first server,

the first server uses the object with the RUBOI it has received from the "naming authority" server.

25. A method of assigning multiple RUBOIs to binary objects, including the steps of:

server that received a binary object checks whether it has an identical object already,

if yes, the server adds RUBOI of the old object to the header of the received message,

else, if the received object does not have a RUBOI yet, the server generates a new RUBOI and assigns it to the object,

when identified based on RUBOIs, binary objects are considered identical if they have at least one pair of identical RUBOIs.

26. A method of alleviating storage of duplicate binary objects, in a Usenet system, the method including:

allocating a Reliable Universal Binary Identifier (RUBOI) to a binary object, using one of the methods claimed in claims 22, 23 , 24 and 25,

determining whether the system has already stored a binary object with such RUBOI,

storing the binary object if the result of step b is negative, and

substituting in the message the binary object by a reference to it and storing the message.

27. A method as claimed in claim 26 where the binary object is text or encoded in text, compressed or uncompressed.

28. A method as claimed in claim 26 or 27, wherein, if the result of step b is positive, the message is stored together with a reference to the already stored binary object.

29. A method as claimed in claim 26, 27 or 28, in which the determining step b is executed via NNTP protocol.

30. A Usenet adapted to operate in accordance with the method of any one of claims 22 to 29.

31. A Usenet as claimed in claim 30, adapted to operatively respond to any one or a combination of commands described above as XARTICLE, XBODY, XHAVE, XNEWSNEWS, XBINARY and / or XPOST where RUBOIs are used instead of UBOIs.

32. A method of operating a Usenet in accordance with any one or a combination of commands XARTICLE, XBODY, XHAVE, XNEWSNEWS, XBINARY and / or XPOST as herein disclosed and where RUBOIs are used instead of UBOIs.

33. A method of transferring messages between at least two ANS-complaint news servers, being a receiving server and a sending server, the method including the steps of:

forwarding attachment identification information (where RUBOIs are used for identification) along with the message identification information (Message ID) to a receiving server,

the receiving server determining whether the attached binary object is to be transferred in accordance with establishing whether the receiving server has a copy of the binary object already, and

if the receiving server does have already a copy of the binary object, indicating to the sender server that no transfer of the binary attachment is necessary but that transferring only the textual part of the message is required.

34. A method as claimed in claims 33, in which when only the textual part of the message is sent, the textual part is stored together with a reference to the already stored binary object.

35. A Usenet adapted to operate in accordance with any one of claims 30 to 33.

36. An identification request broadcast method as herein disclosed.

37. A recognition event broadcast method as herein disclosed.

38. A centralised identification method as herein disclosed.

39. A multiple reliable identifier method as herein disclosed.

40. A XARTICLE command as herein disclosed.

41. A XBODY command as herein disclosed.

42. A XHAVE command as herein disclosed.

43. A XNEWSNEWS command as herein disclosed.

44. A XBINARY command as herein disclosed.

45. A XPOST command as herein disclosed.

46. A method as herein disclosed.

47. A Usenet/system/apparatus as herein disclosed.

48. A method as claimed in any one of claims 1-5 or 13-19, where the UBOI is a RUBOI.

49. A Usenet as claimed in any one of claims 6, 10-12, 20 or 21, where the UBOI is a RUBOI.

50. A method of coordinating the identification of objects with their associated descriptions (metadata) in a newsgroup of the Usenet, the method including the steps of:

generating a first tag, the first tag being readable in a manner for the purposes of identifying a description,

attaching the first tag to a message containing the description,

determine from the first tag, a second tag, the second tag being adapted to identify an object,

attaching the second tag to the message containing the object,

posting the messages.

51. A method of downloading messages from the Usenet, the method including the steps of:

receiving headers of messages available for downloading,

scanning these headers to identify which messages contain descriptions,

downloading the messages containing descriptions,

representing the descriptions to the user to make a decision regarding the downloading of associated objects,

if the user wants to download an associated object,

reading a first tag associated with the description, generating a second tag adapted to identify an object,

scanning the headers in order to locate a tag equivalent to the second tag, and

downloading the message having the located tag.

52. A method as claimed in claim 50 or 51, wherein the second tag is the same as the first tag.

DATED THIS 31st day of December, 1999

CSIRO, DIVISION OF MATHEMATICAL AND INFORMATION SCIENCES

WATERMARK PATENT & TRADEMARK ATTORNEYS
290 BURWOOD ROAD
HAWTHORN VICTORIA 3122
AUSTRALIA

RCS/SH

Changes to Storage

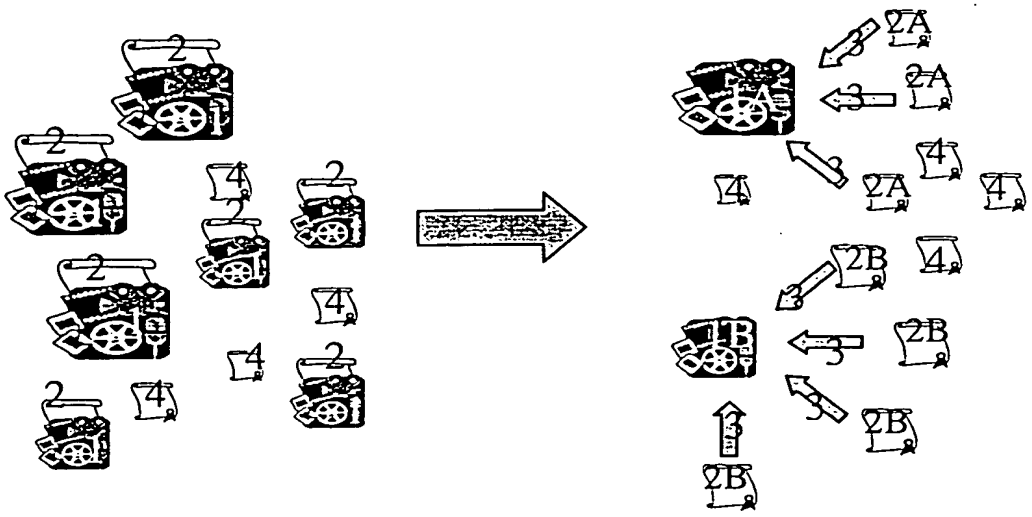
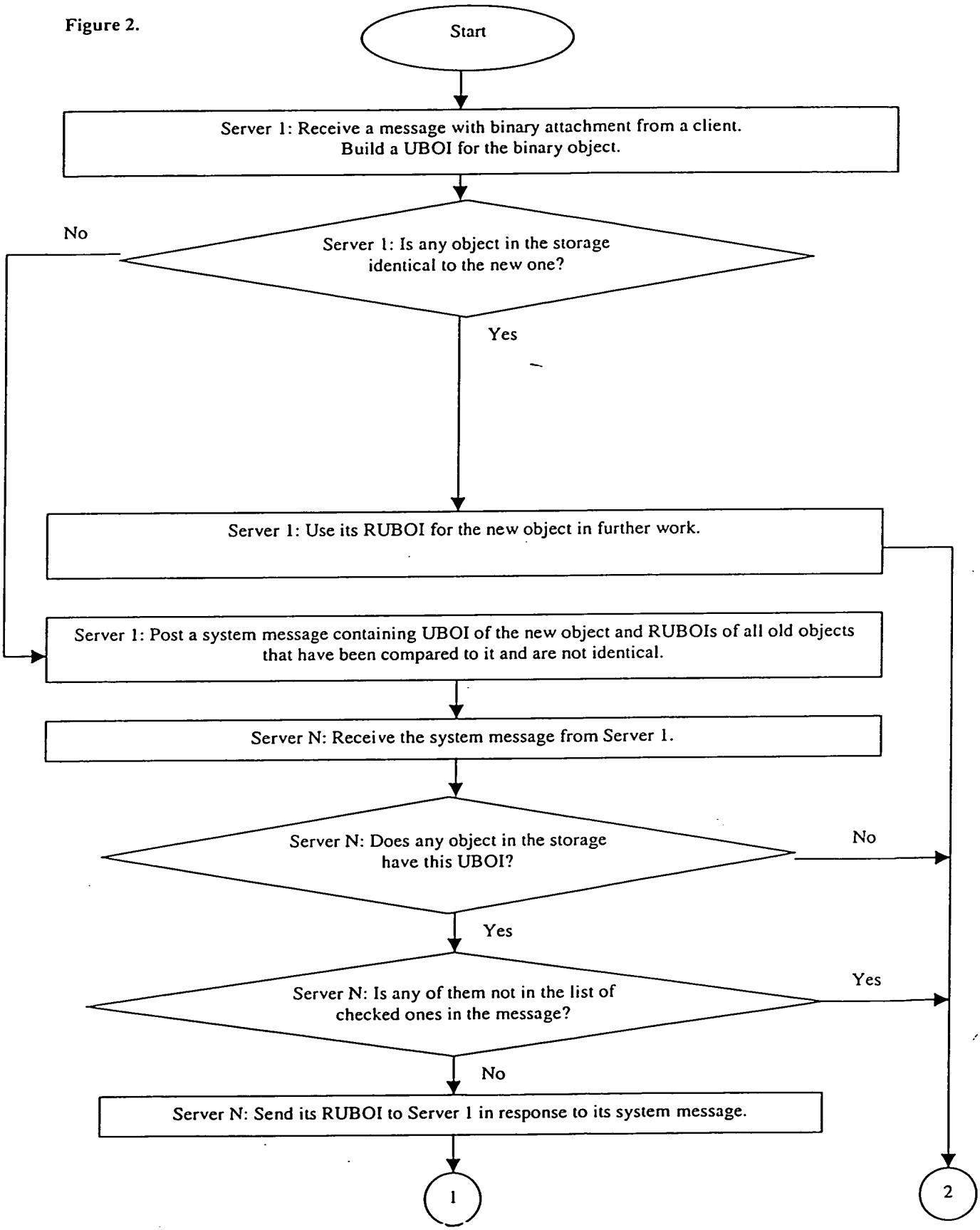


Figure 1.

Figure 2.


Start
Server 1: Receive a message with binary attachment from a client.
Build a UBOI for the binary object.
Server 1: Is any object in the storage
identical to the new one?
No
Yes
Server 1: Use its RUBOI for the new object in further work.
Server 1: Post a system message containing UBOI of the new object and RUBOIs of all old objects
that have been compared to it and are not identical.
Server N: Receive the system message from Server 1.
Server N: Does any object in the storage
have this UBOI?
No
Yes
Server N: Is any of them not in the list of
checked ones in the message?
Yes
No
Server N: Send its RUBOI to Server 1 in response to its system message.
1
2

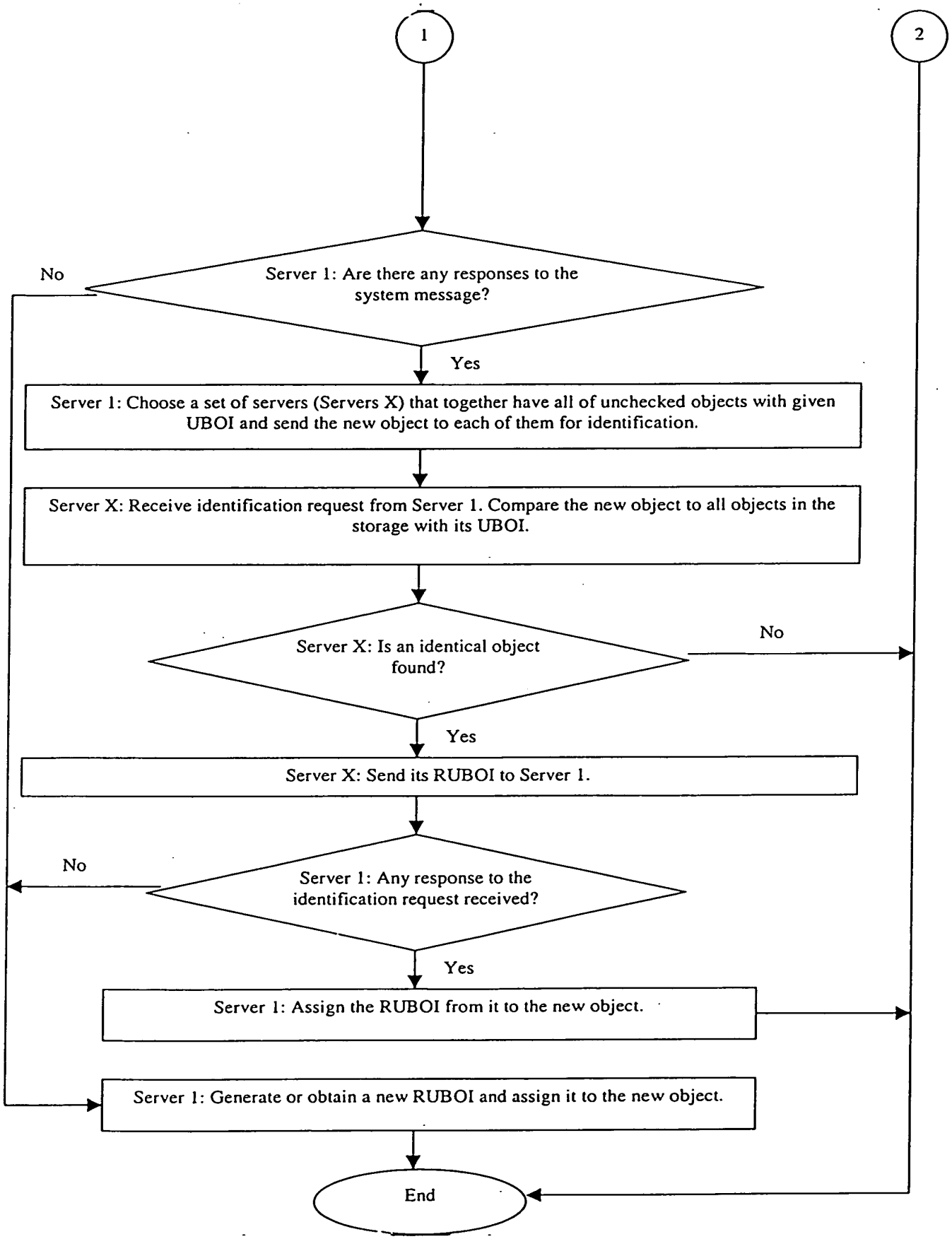

1
2
Server 1: Are there any responses to the system message?
No
Yes
Server 1: Choose a set of servers (Servers X) that together have all of unchecked objects with given UBOI and send the new object to each of them for identification.
Server X: Receive identification request from Server 1. Compare the new object to all objects in the storage with its UBOI.
Server X: Is an identical object found?
No
Yes
Server X: Send its RUBOI to Server 1.
Server 1: Any response to the identification request received?
No
Yes
Server 1: Assign the RUBOI from it to the new object.
Server 1: Generate or obtain a new RUBOI and assign it to the new object.
End

Figure 3.

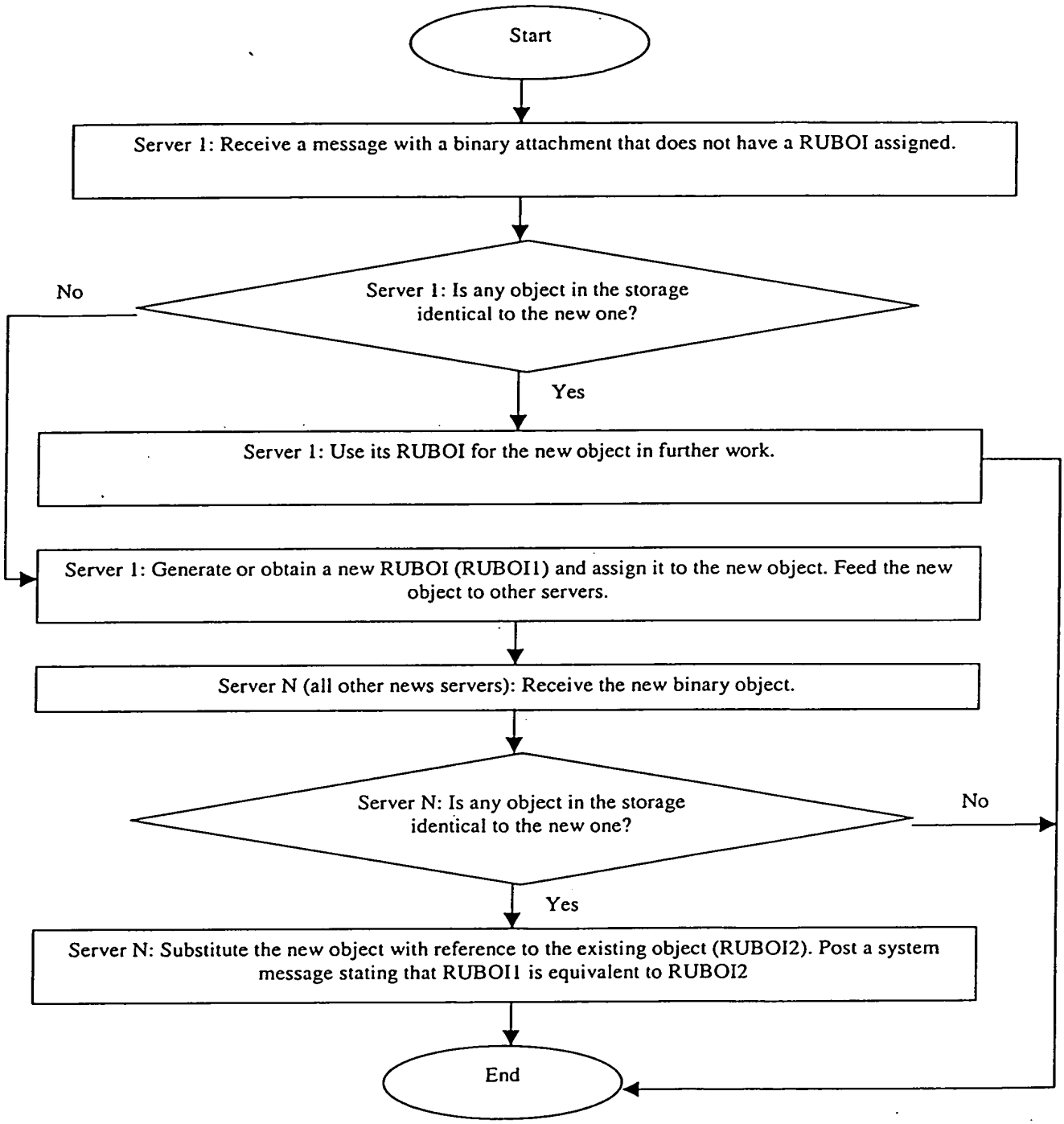
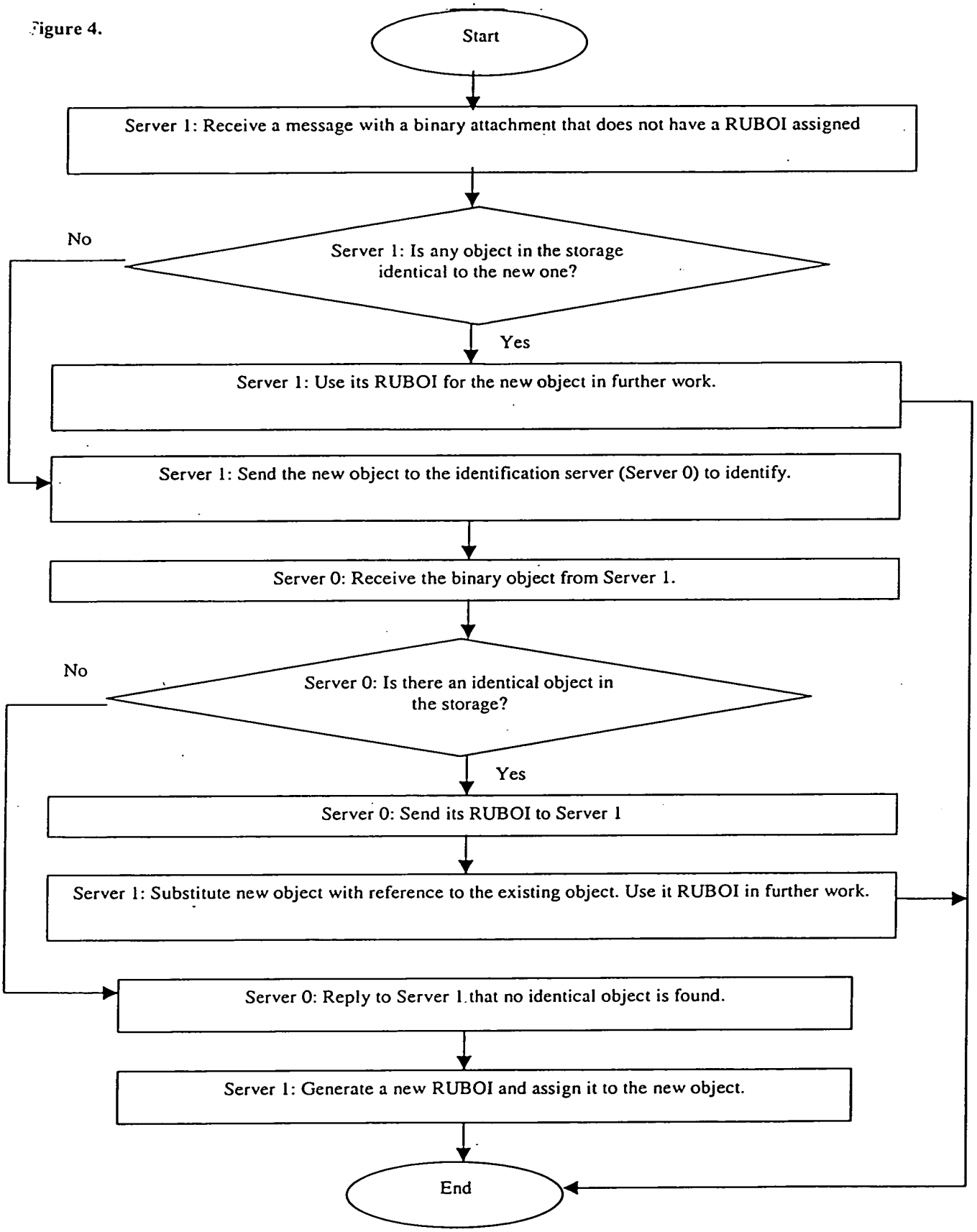

Start
Server 1: Receive a message with a binary attachment that does not have a RUBOI assigned.
Server 1: Is any object in the storage identical to the new one?
No
Yes
Server 1: Use its RUBOI for the new object in further work.
Server 1: Generate or obtain a new RUBOI (RUBOI1) and assign it to the new object. Feed the new object to other servers.
Server N (all other news servers): Receive the new binary object.
Server N: Is any object in the storage identical to the new one?
No
Yes
Server N: Substitute the new object with reference to the existing object (RUBOI2). Post a system message stating that RUBOI1 is equivalent to RUBOI2
End

Figure 4.


Start
Server 1: Receive a message with a binary attachment that does not have a RUBOI assigned
Server 1: Is any object in the storage identical to the new one?
No
Yes
Server 1: Use its RUBOI for the new object in further work.
Server 1: Send the new object to the identification server (Server 0) to identify.
Server 0: Receive the binary object from Server 1.
Server 0: Is there an identical object in the storage?
No
Yes
Server 0: Send its RUBOI to Server 1
Server 1: Substitute new object with reference to the existing object. Use it RUBOI in further work.
Server 0: Reply to Server 1 that no identical object is found.
Server 1: Generate a new RUBOI and assign it to the new object.
End